判断：

1、java的源代码中定义为几个类，编译结果就生成几个.class字节码文件 **（对）**

2、java源程序由类定义组成，每个程序可以定义若干个类，但只有一个类是主类 **（对）**

3、若源程序文件以 A.java 命名，编译后只生成一个名为A的字节码文件 **（错）** 改为： 当一个Java源程序文件以A.java命名并编译成功后，会生成一个与类名同名，扩展名为.class的字节码文件，即A.class

4、java程序是运行在java虚拟机（JVM）中的 **（对）**

5、Java 程序对计算机硬件平台的依赖性很低 **（对）**

6、Java 可以用来进行多媒体及网络编程 **（对）**

7、Java 语言具有较好的安全性、可移植性及与平台无关等特性 **（对）**

8、java语言的源程序不是编译型的，而是编译解释型的 **（对）**

9、java Application 程序中，必有一个主方法main()，该方法有没有参数都可以 **（错）**

解释：参数是String[] args 即public static void main(String[] args)

填空

目前Java 应用开发中的三个平台版本分别是 **Java SE**、**Java ME**、**Java EE**

Java 源程序文件编译后产生的文件称为**字节码文件**，其拓展名为 **.class**

java的编译源代码的命令是**javac**，执行字节码文件运行程序的命令是**java**

选择

1、main 方法是 Java Application 程序执行的人口点，关于 main 方法正确的是 **public static void main(String[] args)**

2、main方法返回的类型是**void**，不能有返回值

3、java程序的执行过程用到一系列JDK工具，其中java.exe是指 **java解释器**

4、在Java 中，负责对字节代码解释执行的是 **java虚拟机**

5、java语言的 **标识符是区分大小写的**

6、源文件名与 public 类名 **必须完全相同**

7、源文件中public类的数目**只有一个**

判断

1.default 语句在 switch分支选择结构中是必须的。**错误**

2.break 语句在 switch 分支选择结构中是必须的。**错误**

3.while 循环中的循环体至少执行一次。**错误**

4.break 语句只是用来结束当次循环。**错误**

break 语句不仅用于结束当前循环，还用于跳出 switch 语句或 for 循环等其他控制结构。当 break 被执行时，它会立即跳出最内层的循环或 switch 语句块。

填空

1、顺序结构、选择结构和**循环结构**是结构化程序设计的3种基本流程控制结构

2、每一个 else 子句都必须和离它最近的**if**子句相对应。

3、循环包括 for 循环、do...while 循环和**for**循环

4、**continue**语句的功能是：跳过循环体内部下面未执行的语句，回到循环体开始位置继续下次循环。

例如：

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) {
        continue;
    }
    System.out.println(i);
}
```

i是从1到10，如果i是偶数，则循环体中System.out.println(i)这一语句就会被跳过，程序回到for循环开头，继续下一次循环（i的值加1），如果i是奇数，则输出i。

那么break语句的功能是？

break 语句主要用于终止它所在的循环结构或者跳出 switch 语句块 例如：

```
for (int i = 1; i <= 10; i++) {
    if (i % 3 == 0) {
        System.out.println("第一个能被3整除的数是：" + i);
        break;
    }
}
```

从1开始检查i是否能被3整除，如果能则输出i，然后遇到了break语句，此时**整个for循环就会终止，不会再检查4-10这些数**。

判断

1、一个数组可以存放许多不同类型的值 **错**

2、数组索引通常是 foat 类型**错**

（数组索引必须是整数，它表示的是数组中某个元素的位置，可以是int或long）

3、如果将单个数组元素传递给方法，并在方法中对其修改，则在被调用方法结束运行时，该元素中存储的是修改后的值。**对**

（如果将数组元素传递给方法并修改它，那么方法结束后，原数组中该元素的值会发生变化，在 Java 中，数组传递的是引用）

java中数组的多维声明必须使用new关键字时指定每个维度的大小，正确的做法是：`int a[][]= new int[10][10];` 声明和初始化是分开的，不能把10写到等号前面去

4、 int a[][]= new int[10,10];**错**

5、 int a[10][10]= new int[][];**错**

6、int a[][]= new int[10][10];**对**

7、int[] a[]= new int[10][10];**对** int[] a[] 和 int a[][] 在语法上是等价的

8、int[][] a= new int[10][10];**对** 是声明二维数组的另一种写法

在返回值类型不是 void 的方法中，不写 return 语句会发生错误吗?可以在返回值类型为 void 的方法中写 return 语句吗

答：会出错，返回值不是void那就必须有return语句，且 return 语句必须返回一个与声明类型兼容的值；可以在void方法中写return语句，但return语句没有返回值，这种return用来提前结束方法的执行

判断

1、在定义一个类的时候，如果类的成员被private 修饰，该成员不能在类的外部被直接访问。**对**

2、Java 中的每个类都至少有一个构造方法，如果一个类中没有定义构造方法，系统会自动为这个类创建一个默认的无参构造方法。**对**

3、声明构造构方法时，不能使用 private 关键字修饰。==**错**==

4、类中static修饰的变量或方法，可以使用类名或对象的引用变量直接访问。==**对**==

5、方法的内部类不能访问外部类的成员变量。**错误**

（方法中的内部类是可以访问外部类的成员变量的，前提是外部类的成员变量必须是 final 或 effectively final）

6、类中的构造方法只有一个。**错误**

7、使用运算符 new 创建对象时，赋给对象的值实际上是一个引用值 **对**

8、对象可以作为方法参数，对象数组不可以作为方法参数。**错** java支持将任何类型的对象作为参数传递给方法，包括数组对象。

填空

1、面向对象的三大特征：**封装、继承、多态** 2、java语言使用关键字**new**来创建类的实例对象

3、定义在类中的变量称为**成员变量**，定义在方法中的变量称为 ==**局部变量**==

4、在类中利用关键字 ==**static**== 修饰的成员变量可以被所有的实例对象共享

==static 修饰的成员变量称为类变量，它不属于某个具体的实例，而是属于类本身，所有实例共享同一个类变量。==

5、类的访问权限关键字包括 **public、protected、private**和默认（default）4种

public：该类对所有其他类可见。

protected：该类对同一包内的类以及子类可见。

private：该类只能在定义它的类内部访问。

default（默认）：包访问权限，可以在同一包内访问

6、java源文件只能有**一个public类**，其他类的个数不限

java中的实例就是对象，说法不同而已

选择

1、下面哪一个是正确的类声明

public void HH {} **错** void是修饰方法的

public class Move() {} **错** 没括号

public class void number {} **错** void

publie class Car {} **对** public class 类名 {}

2、在以下哪种情况中，构造方法会被调用?

类定义

创建对象 **对**

调用对象方法

使用对象的变量时

3、下面对于构造方法，正确的有哪些?(多选)

方法名必须和类名相同 **对**

方法名前没有返回值类型的声明 **对**

在方法中==不能==使用 return 语句返回一个值 **对** 构造方法不返回任何值

当定义带参数的构造方法，系统默认的不带参数的构造方法依然存在。**错**

构造方法示例：

```
public class Person {
    String name;
    int age;

    // 无参构造方法
    public Person() {
        System.out.println("默认构造方法");
    }

    // 有参构造方法
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

4、使用 this 调用类的构造方法，下面说法正确的是?(多选)

A.使用 this 调用构造方法的格式为 this([参数 1，参数 2,...]) **对**

this() 语句用于在构造方法中调用本类的其他构造方法，括号中的参数列表与目标构造方法的参数列表必须匹配

B.只能在构造方法中使用 this 调用其他的构造方法 **对**

this() 关键字只能在构造方法内部使用，用于调用本类的其他构造方法，==且this() 语句只能出现在构造方法的第一行==

C.使用 this 调用其他构造方法的语句必须放在第一行 **对**

D.不能在一个类中的两个构造方法中使用 this 互相调用 **错** ==可以互相调用的==

例子：

```
public class test {
    String name;
    int age;

    // 无参构造方法
    public test() {
        this("无名氏", 18); // 调用有参构造方法
        System.out.println("无参构造方法");
    }

    // 有参构造方法
    public test(String name, int age) {
        this.name = name;
        this.age = age;
        System.out.println("有参构造方法");
    }
```

```
    public static void main(String[] args) {
        test test1 = new test();
        test test2 = new test("张三", 20);
}
}
```

结果： 有参构造方法

无参构造方法

有参构造方法

5、下面哪些可以使用 static 关键字修饰?(多选)

static 修饰的成员变量称为类变量，属于类本身，而不是类的实例。

A.成员变量 **对**

B.局部变量

C.成员方法 **对** static 修饰的方法是静态方法，属于类本身，可以通过类名直接访问

D.成员内部类 **对**

成员内部类也可以被 static 修饰，称为静态内部类。静态内部类与外部类的实例对象无关，不能直接访问外部类的实例变量和方法，只能访问外部类的静态成员。

什么叫"static 修饰的方法是静态方法，属于类本身，可以通过类名直接访问"？

```
public class MyClass {
    // 静态方法
    public static void staticMethod() {
        System.out.println("这是一个静态方法! ");
    }

    // 普通成员方法
    public void instanceMethod() {
        System.out.println("这是一个实例方法! ");
    }

    public static void main(String[] args) {
        // 通过类名直接调用静态方法
        MyClass.staticMethod();  // 无需创建对象，直接通过类名调用

        // 需要通过实例来调用实例方法
        MyClass obj = new MyClass();
        obj.instanceMethod();  // 必须创建对象后才能调用实例方法
    }
}
```

static 修饰的方法是静态方法，属于类本身，可以通过类名直接访问。 这句话怎么理解

```
public class MyClass {
    // 静态方法
    public static void staticMethod() {
        System.out.println("这是一个静态方法! ");
    }

    // 普通成员方法
    public void instanceMethod() {
        System.out.println("这是一个实例方法! ");
    }

    public static void main(String[] args) {
        // 通过类名直接调用静态方法
        MyClass.staticMethod();  // 无需创建对象，直接通过类名调用

        // 需要通过实例来调用实例方法
        MyClass obj = new MyClass();
        obj.instanceMethod();  // 必须创建对象后才能调用实例方法
    }
}
```

能够正确导人包 lib 中类 Circle 的语句是? **import lib.Circle;**

访问定义在默认权限类中的 private 成员，下列访问形式中的正确的是?

A.在本类中访问 **对**

B.在同一文件的类中访问

C.在同一包的类中访问

D.在不同包的类中访问

判断

1.抽象方法必须定义在抽象类中 **对**，所有抽象类中的所有方法都是抽象方法 **错**

抽象类中可以定义非抽象方法

2.Java 中被 final 关键字修饰的变量，不能被重新赋值。 **对**

3.不存在继承关系的情况下，可以实现方法的重写。**错**

重写是子类对父类（或实现接口）的方法进行重新实现

4.接口中只能定义常量和抽象方法。**错** ==java8之后，还有默认方法和静态方法==

5.方法的内部类不能访问外部类的成员变量 **错**

==方法内部的类也可以访问外部类的实例变量和静态变量==

```
public class OuterClass {
    private int instanceVar = 10;  // 外部类的实例变量

    public void method() {
        int localVar = 20;  // 方法的局部变量

        // 方法的内部类
        class InnerClass {
            void display() {
                System.out.println("Outer class instance variable: " +
instanceVar);  // 访问外部类的实例变量
                System.out.println("Method local variable: " + localVar);  // 访问
方法的局部变量
            }
        }

        InnerClass inner = new InnerClass();
        inner.display();
    }

    public static void main(String[] args) {
        OuterClass outer = new OuterClass();
        outer.method();
    }
}
```

1.关于 super 关键字以下说法哪些是正确的?(多选):

==A.super 关键字可以调用父类的构造方法== **对**

==B.super 关键字可以调用父类的普通方法== **对**

==C.super与this不能同时存在于同一个构造方法中== **对**

D.super 与 this 可以同时存在于同一个构造方法中

super()用于调用父类的构造方法，需要写在子类构造方法的第一行，this()用于调用当前类的其他构造方法，也需要写在第一行，都需要写在第一行，所以不能出现在同一个构造方法中

但是super和this有其他用途，比如：

```
public class Child extends Parent {
    private int value;

    public Child(int value) {
        super(); // 调用父类的无参构造方法，必须放在第一行
        this.value = value; // 使用 this 引用当前对象的实例变量
```

```
    }
}
```

这时是可以方法一个构造方法里面的，只是this()和super()不能放在同一个构造方法里面。

==题描述不清楚==

2.在 Java 语言中，以下说法哪些是正确的(多选)

==A.一个类可以实现多个接口== **对**

==B.不允许一个类继承多个类== java是单继承 **对**

==C.允许一个类同时继承一个类并实现一个接口== **对**

==D.允许一个接口继承另外一个接口== **对**

3.关于抽象类，下列哪些说法是正确的?(多选)

==A.抽象类中可以有非抽象方法==**对**

B.如果父类是抽象类，则子类必须重写父类的所有抽象方法

错误，子类如果是一个非抽象类，那它需要重写父类的所有抽象方法，子类是抽象类时则不用

==C.不能用抽象类去创建对象==**对**

抽象类不能直接实例化，因为它包含抽象方法，缺少完整的实现。**只能通过子类来实例化**（只要子类实现了所有的抽象方法）。

例子：

```
abstract class Animal {
    abstract void sound();
}

// Animal animal = new Animal();  // 错误：不能实例化抽象类

class Dog extends Animal {
    void sound() {
        System.out.println("Bark");
    }
}

Animal dog = new Dog();  // 正确：可以实例化子类
```

D.接口和抽象类是同一个概念

抽象类的作用？

定义通用的框架和模板，在其中定义一些通用的行为（方法）和属性（成员变量），供子类继承和扩展。封装共有的行为，强制子类实现特定方法。 抽象类中的抽象方法是子类必须实现的

下面关于内部类，哪些说法是正确的?(多选)

==A.成员内部类是外部类的一个成员变量，可以访问外部类的其他成员== **对**

（成员内部类就是类里面的类，成员内部类就像外部类的一个普通成员一样，可以访问外部类的所有成员，包括私有成员。）

B.外部类可以访问成员内部类的成员

（外部类不能直接访问成员内部类的成员，**除非通过成员内部类的对象来访问**，也就是在本类中定义一个method，这个method创建内部类实例，访问成员内部类的成员，其他类通过外部类对象来调用这个method以访问成员内部类的成员）

==C.方法内部类只能在其定义的当前方法中进行实例化==**对**

（方法内部类是定义在方法中的类。方法内部类只能在它所在的方法内被实例化，不能在方法外部访问。方法内部类不能有访问修饰符，也不能声明为static。）

==D.静态内部类中可以定义静态成员，也可以定义非静态成员==**对**

```
class Outer {
    void outerMethod() {
        class Inner {
            void print() {
                System.out.println("Inside method");
            }
        }

        Inner inner = new Inner();  // 方法内部类实例化
        inner.print();
    }
}

class Test {
    public static void main(String[] args) {
        Outer outer = new Outer();
        outer.outerMethod();
    }
}
```

Outer 类中定义了一个成员内部类 Inner,需要在 main()方法中创建 Inner 类实例对 象，下面哪个是正确的?

A.Inner in = new Inner( ) **错误**

成员内部类是依赖于外部类的实例的，因此无法直接通过 new Inner() 来创建对象。必须通过外部类的实例来创建内部类的实例。

B. Inner in = new Outer. Inner( )

C.Outer.Inner in = new Outer. Inner( )

==D. Outer. Inner in = new 0uter( ).new Inner()== **正确**